Crittografia post quantistica (PQC)

Crittografia

Luciano Margara

Unibo

2026

Dato di fatto

- Nel 2024, il NIST (National Institute of Standards and Technology) ha pubblicato una suite di standard post-quantistici per meccanismi di incapsulamento di chiavi e schemi di firma digitale.
- Questi schemi sono progettati per sostituire RSA ed ECC che sono vulnerabili agli attacchi quantistici. Si prevede che Kyber (KEM) e Dilithium (firma) saranno i più adottati nei prossimi anni.

key encapsulation mechanism

Perché ?

Computer Quantistici

Concetto introdotto da Yuri Manin (1980) e Richard Feynman (1981). I computer quantistici usano fenomeni della meccanica quantistica: Superposition, Interference e Entanglement.

Qubit:
un qubit è l'analogo quantistico di un bit. Può trovarsi in due stati contemporaneamente, ciascuno con una certa probabilità. Un registro di n qubit può rappresentare 2^n stati simultaneamente. Una funzione $f : \{0,1\}^n \to \{0,1\}^n$ può essere valutata simultaneamente su tutti i 2^n input. Alla misurazione, il registro collassa in uno solo degli stati, secondo la distribuzione di probabilità.

Algoritmo di Shor

I protocolli a chiave pubblica attualmente utilizzati basano la loro sicurezza sulla complessità computazionale di alcuni problemi matematici: fattorizzazione di interi, logaritmo discreto nei gruppi finiti, logaritmo discreto su curve ellittiche.

L'algoritmo di Shor (1994) è un algoritmo quantistico efficiente (tempo polinomiale) per risolvere questi problemi.

Quando saranno costruiti i computer quantistici?

- 1998: Jones & Mosca
  computer quantistico da 2 qubit.
- 2017: IBM - 50 qubit
  `tinyurl.com/IBMqc50`
- 2019: Google - 53 qubit
  `tinyurl.com/GoogleQC`
- 2021: IBM - 127 qubit
  `tinyurl.com/IBMqc127`
- 2022: IBM - 433 qubit
  `tinyurl.com/IBMqc433`
- 2023 (dicembre): IBM - 1.121 qubit
  `tinyurl.com/IBMqc1121`

Computer quantistici tolleranti agli errori

- I computer quantistici odierni non sono "fault tolerant".
- Attualmente, i qubit fisici sono troppo instabili.
- Si lavora per combinare molti qubit fisici in uno logico, resistente agli errori.
- Stima Gidney & Ekera (2021): per fattorizzare RSA-2048 tramite l'algoritmo di Shor occorrono 6.000 qbit logici ovvero 20M qubit fisici e 8 ore di calcolo.
- Dicembre 2023: Harvard/MIT/QuEra: progresso notevole nella correzione degli errori.
- Non ci sono ostacoli teorici alla costruzione di computer quantistici fault-tolerant.

Computer quantistici tolleranti agli errori

- I computer quantistici costruiti finora non rappresentano ancora una minaccia per la crittografia attuale.
- È ancora troppo presto per prevedere quando verranno costruiti computer quantistici tolleranti agli errori su larga scala.
- Il prossimo traguardo sarà costruire un singolo qubit logico funzionante.

La minaccia dei computer post quantistici

Cos'è un attacco HNDL?

- Attacchi HNDL (Harvest Now, Decrypt Later): gli avversari (es. agenzie governative) raccolgono e archiviano grandi volumi di traffico cifrato oggi.
- Quando i computer quantistici saranno disponibili, potranno decifrare questi dati.
- Colpisce in particolare cifrari basati su RSA, DH ed ECC.

Domande:

- Cosa possiamo fare per mitigare questa minaccia?
- Quando dobbiamo agire?
  - Ora?
  - Tra 5 anni?
  - Tra 10? 20?

NSA - IAD: Annuncio di transizione (agosto 2015)

IAD = Information Assurance Directorate ovvero una sezione interna della National Security Agency (NSA) degli Stati Uniti.

- Inizieremo la transizione verso algoritmi resistenti ai computer quantistici nel prossimo futuro.
- Intendiamo pianificare e comunicare in anticipo, collaborare con USG, vendor, enti di standardizzazione, garantire una transizione aperta e trasparente.
- Obiettivo: sicurezza efficace ed economicamente sostenibile contro i futuri computer quantistici.

Fase transitoria: Fino a quando la nuova suite non sarà sviluppata e i prodotti disponibili, continueremo a usare gli algoritmi attuali.

Standardizzazione NIST della crittografia post-quantistica

- 30 novembre 2017: 69 proposte ricevute (Round 1)
- 30 gennaio 2019: 26 selezionate per il Round 2
- 22 luglio 2020: 7 + 8 selezionate per il Round 3
- 5 luglio 2022:
  - Kyber selezionato come KEM
  - Dilithium, Falcon, SPHINCS+ come schemi di firma
  - LMS e XMSS (firma) già standardizzati (SP 800-28)
- 13 agosto 2024:
  - FIPS 203: Kyber
  - FIPS 204: Dilithium
  - FIPS 205: SPHINCS+
  - FIPS 206 (Falcon) atteso a breve

CNSA - Commercial National Security Algorithm Suite 2.0

Obiettivo: raccomandare algoritmi resistenti ai computer quantistici per la protezione di: informazioni classificate, sistemi di sicurezza nazionale e applicazioni commerciali ad alta sicurezza. Data di pubblicazione: 7 settembre 2022.
Algoritmi raccomandati:

- Kyber: KEM basato su reticoli
- Dilithium: firma digitale basata su reticoli
- SPHINCS+: firma digitale stateless basata su hash
- AES-256 e SHA-384: mantenuti per crittografia simmetrica e hash

CNSA 2.0 sostituisce CNSA 1.0 (che includeva RSA e ECC), supporta la transizione alla crittografia post-quantistica ed è applicabile fino al 2030 e oltre

Adozione della crittografia post-quantistica

- Apple: PQ3 (2024)
  Introdotto in iMessage su iOS/macOS. Combina ECC con Kyber (ML-KEM). Protezione contro attacchi "Harvest Now, Decrypt Later". Formalmente verificato.
- Signal: PQXDH (2023)
  Estensione post-quantistica del protocollo X3DH. Utilizza Kyber per la protezione della fase di handshake iniziale.
- Google, Amazon e altri
  Non hanno ancora adottato PQC nei prodotti pubblici, ma partecipano attivamente alla standardizzazione (es. NIST PQC). In fase di test e sperimentazione.

Prerequisiti matematici: $\mathbb{Z}_p[x]$

p è un numero primo > 2

$\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$

$\mathbb{Z}_p[x]$ è l'insieme dei polinomi con coefficienti in $\mathbb{Z}_p$ (riduce i coefficienti tra 0 e p-1)

Esempio: Sia $p = 7$,

$f(x) = 5 + 4x^2 + 3x^3 \in \mathbb{Z}_7[x]$

$g(x) = 6 + 3x^+2x^2 \in \mathbb{Z}_7[x]$

abbiamo:

$f(x) + g(x) = 4 + 3x + 6x^2 + 3x^3$

$f(x) - g(x) = 6 + 4x + 2x^2 + 3x^3$

$f(x)g(x) = 2 + x + 6x^2 + 2x^3 + 3x^4 + 6x^5$

Prerequisiti matematici: $\mathbb{Z}_p[x]/(x^n+1)$

(riduce il grado a x^n-1)

n è un numero intero positivo.
$R_p = \mathbb{Z}_p[x]/(x^n+1)$ è l'anello dei polinomi in $\mathbb{Z}_p[x]$ di grado minore di n, con la moltiplicazione di polinomi eseguita modulo la riduzione polinomiale x^n+1.
Quindi per moltiplicare $f(x)$ e $g(x)$ in R_p:

1. Si calcola $h(x) = f(x)g(x)$ in $\mathbb{Z}_p[x]$, $deg(h(x)) \leq 2n-2$
2. Si calcola $r(x)$ come il resto della divisione di $h(x)$ per x^n+1, $deg(r(x)) \leq n-1$.
3. si pone $f(x)g(x) = r(x)$.

Prerequisiti matematici: $\mathbb{Z}_p[x]/(x^n+1)$

Esempio:

$p = 41$, $n = 4$

$f(x) = 32 + 17x^2 + 22x^3$

$g(x) = 11 + 7x + 19x^2 + x^3$

$h(x) = 24+19x+16x^2+24x^3+26x^4+25x^5+22x^6$

poniamo $x^4 \to -1$, $x^5 \to -x$, $x^6 \to -x^2$ li sostituiamo perché gradi maggiori di n=4

e semplifichiamo:

$$\begin{aligned} r(x) &= 24 + 19x + 16x^2 + 24x^3 - 26 - 25x - 22x^2 \\ &= 39 + 35x + 35x^2 + 24x^3 \end{aligned}$$

Prerequisiti matematici: $\mathbb{Z}_p[x]/(x^n+1)$

$R_p = \mathbb{Z}_p[x]/(x^n+1)$

$f(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$

$f = (a_0, a_1, \ldots, a_{n-1})$

Esempio:

$R_{41} = \mathbb{Z}_p[x]/(x^4+1)$

$f(x) = 23 + 11x^2 + 7x^3 \rightarrow f = (23, 0, 11, 7)$

$g(x) = 40 + 5x + 16x^2 \rightarrow g = (40, 5, 16, 0)$

$f + g = (22, 5, 27, 7)$

$fg = (12, 3, 29, 7)$

Prerequisiti matematici: Modulo R_p^k

Rp^k = vettore di k elementi (gli elementi sono Polinomi in Rp)

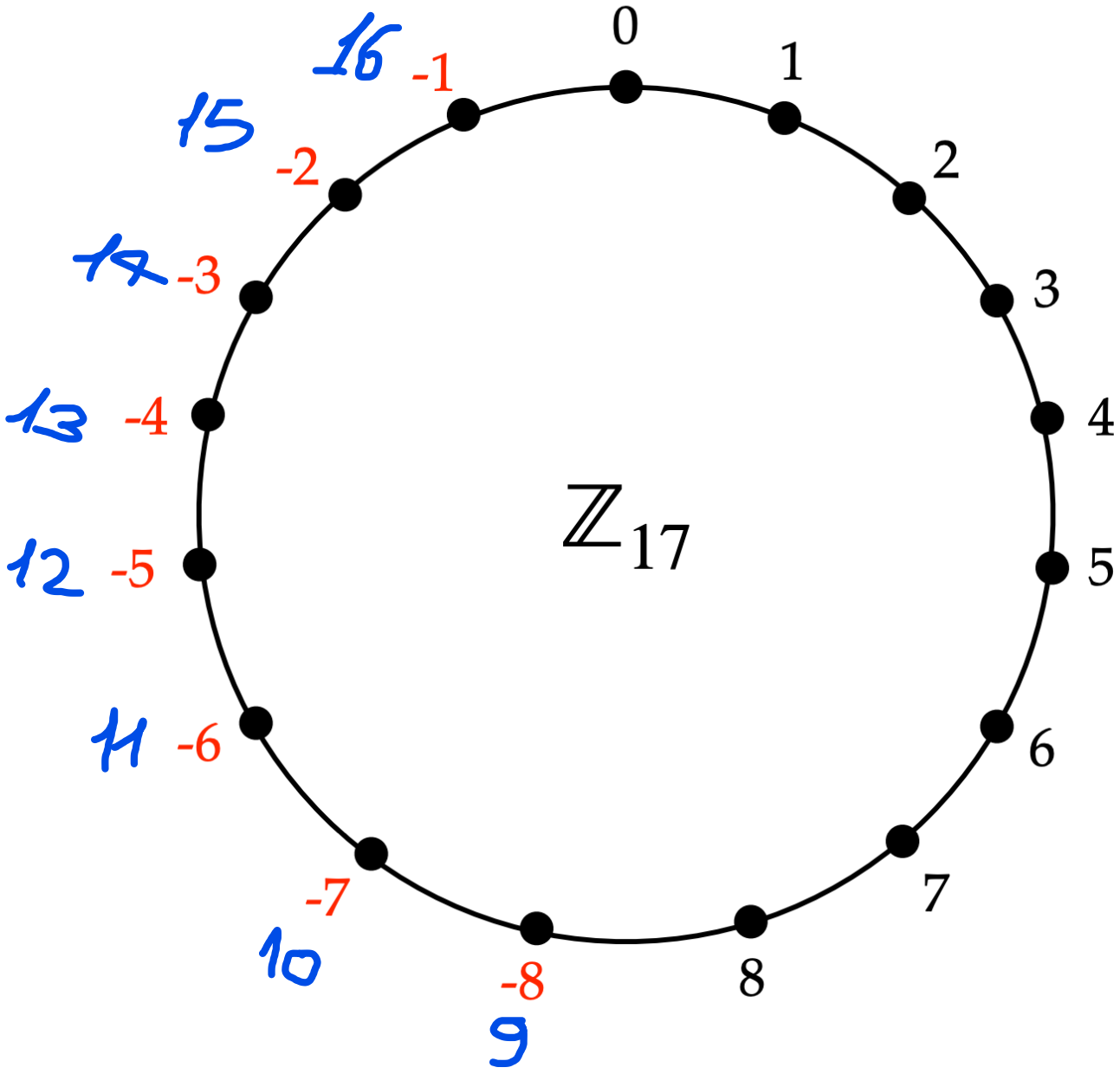
R_p^k è il modulo contenente tutti i vettori di k posizioni con elementi in R_p.

Addizione e sottrazione si calcolano componente per componente.

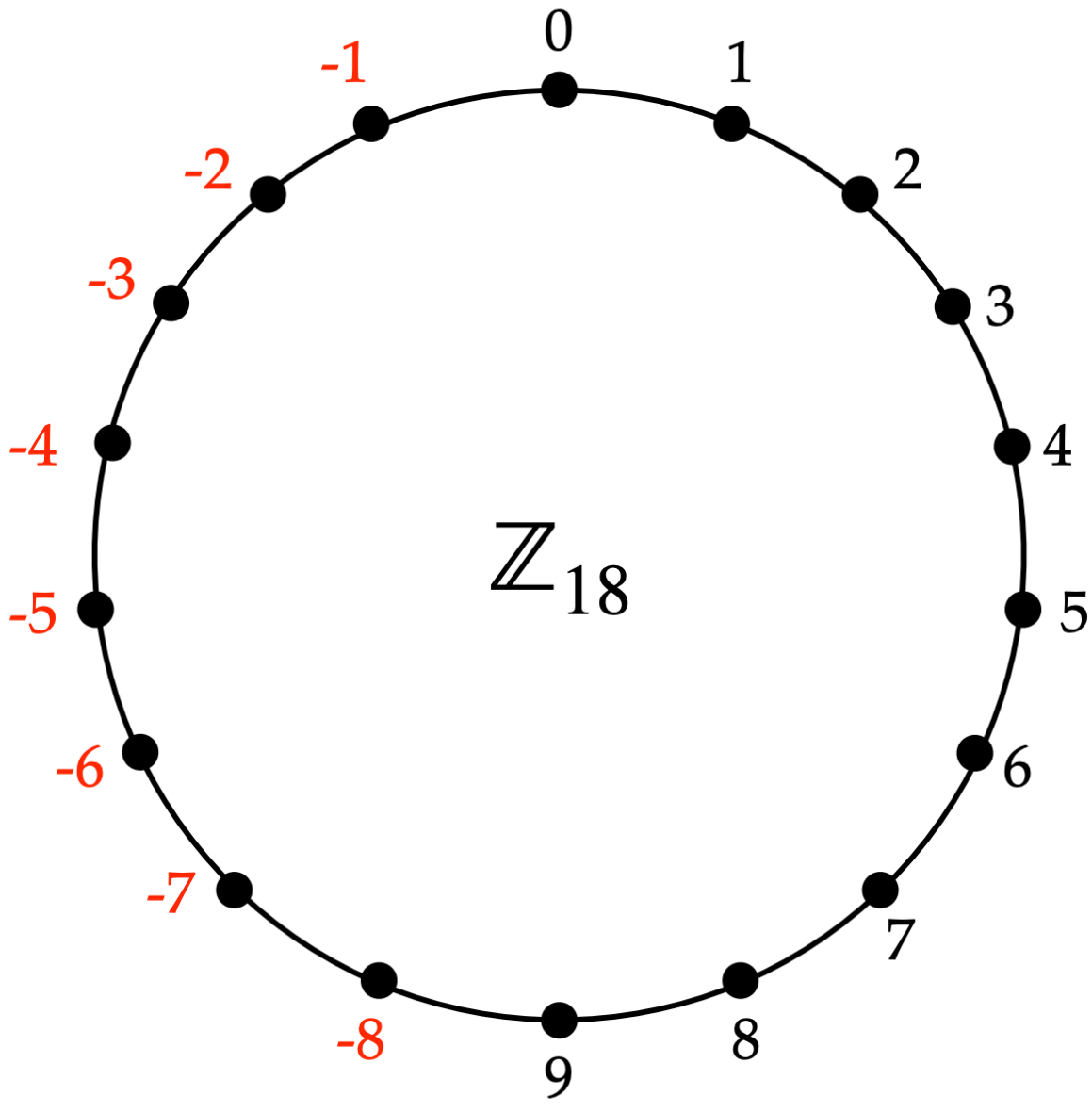
Il prodotto interno di due elementi di R_p^k è un elemento di R_p.

I vettori in R_p^k sono vettori colonna.

Prerequisiti matematici: Modulo simmetrico



Prerequisiti matematici: Modulo simmetrico



Prerequisiti matematici: Arrotondamenti

Sia p un numero primo dispari e $r \in [0, p-1]$.

E sia

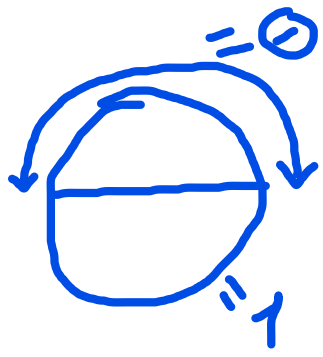
$$r' = r \ mods \ p \in \left[-\frac{p-1}{2}, \frac{p-1}{2} \right]$$

r' fa parte del modulo simmetrico di p

Definiamo:

se r' si trova nella parte vicina allo zero, allora = 0
se r' si trova nella parte lontana allo zero, allora = 1

$$Round_p(r') = \begin{cases} 0, & \text{se } -\frac{p}{4} < r' < \frac{p}{4} \\ 1, & \text{altrimenti} \end{cases}$$



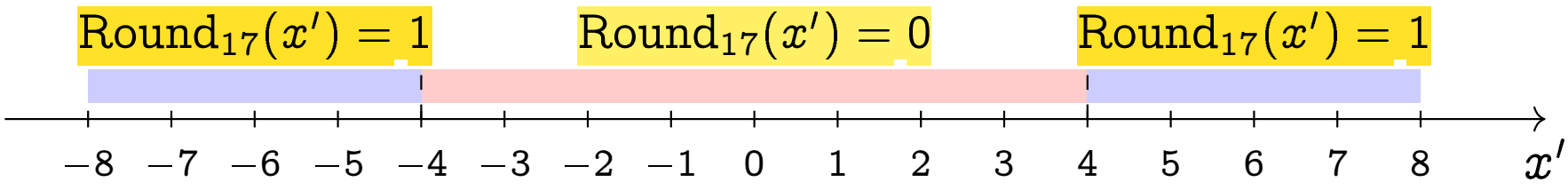
$Round_p$ può essere estesa ai polinomi in R_p applicandola a tutti i coefficienti del polinomio.

Esempio: Sia $p = 3329$, $p/4 = 832.25$:

$$Round_{3329}(3000 + 1500x + 2010x^2 + 37x^3) = x + x^2$$

Prerequisiti matematici: Arrotondamenti

Rappresentazione della soglia di arrotondamento per $p = 17$ e $x' \in [-8, 8]$:



Prerequisiti matematici: Size di oggetti

Esempio (Modulo Simmetrico di p=17):
> r=8 <-> modulo di (8 mods 17) = 8
> r=9 <-> modulo di (9 mods 17) = 8

Modulo del coefficiente r modulo simmetrico di p

$$r \in \mathbb{Z}_p \rightarrow \|r\|_\infty = |r \ mods \ p|$$

$$f = (a_0, a_1, \ldots, a_{n-1}) \rightarrow \|f\|_\infty = \max_{0 \leq i \leq n-1} \|a_i\|_\infty$$

$$v = [f_1, f_2, \ldots, f_k] \rightarrow \|v\|_\infty = \max_{1 \leq i \leq k} \|f_i\|_\infty$$

norma infinito applicata alla lista di polinomi = applicare la norma infinito ad ogni polinomio e prendere il massimo tra quelli (cioé per ogni polinomio fare il passaggio precedente)

norma infinito applicato al polinomio = applicare la norma infinito ad ogni coeff. e prendere il massimo tra quelli

Prerequisiti matematici: Small polynomials

polinomio

$f \in R_p$ è small se $\|f\|_\infty$ è piccola rispetto a $p/2$

Sia η un intero positivo

Definiamo $S_\eta = \{f \in R_p : \|f\|_\infty \leq \eta\}$

di solito si sceglie ben al di sotto di p/2

Esempio: $p = 31$,

$1 + 30x + 29x^2 + x^4 + 2x^5 \in S_2$

Questo polinomio ha come max norma infinito di tutti i coeff uguale a 2, quindi é un Small Polynomial (pk piccola rispetto a p/2)

Quindi, questo polinomio é small rispetto a theta = 2

Esempio: S_1 contiene i polinomi in R_p i cui coefficienti ridotti $mods\ p$ sono $-1, 0, 1$

Prerequisiti matematici: Small polynomials

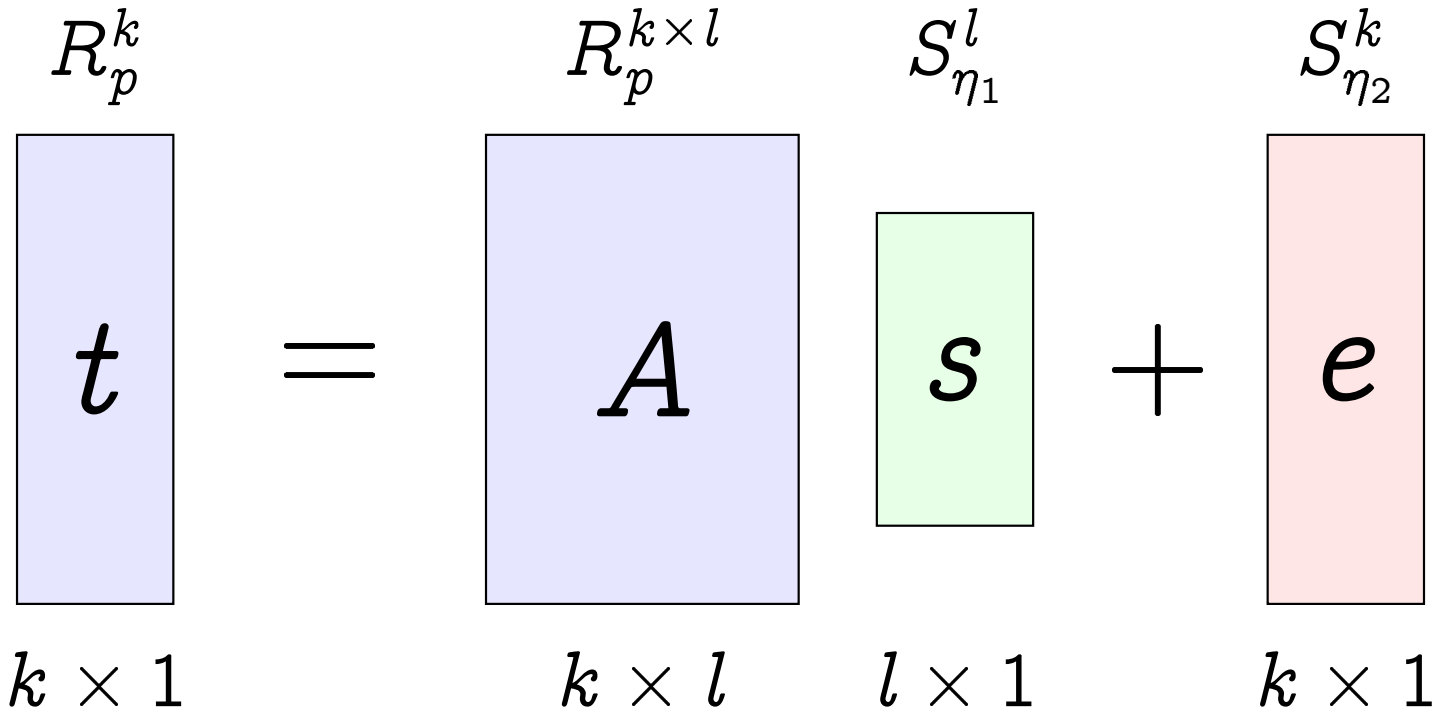
Prodotto di due polinomi piccoli

$$f \in S_{\eta_1}, g \in S_{\eta_2} \implies fg \in S_{n\eta_1\eta_2}$$

Prodotto scalare di due “vettori di polinomi piccoli”

$$v \in S_{\eta_1}^k, w \in S_{\eta_2}^k \implies v^T w \in S_{kn\eta_1\eta_2}$$

Module Learning With Errors (MLWE)



Notazione

Useremo $a \in_R A$ per indicare il processo di generazione casuale di un elemento a appartenente all'insieme A.

Module Learning With Errors (MLWE)

Parametri del problema:

p primo

n, k, l interi con $k \geq l$

η_1, η_2 interi molto più piccoli di $p/2$

Input: (A, t) ottenuti come segue

$A \in_R R_p^{k \times l}$ (dove $R_p = \mathbb{Z}_p[x]/(x^n + 1)$)

$t = As + e$ con $s \in_R S_{\eta_1}^l$, $e \in_R S_{\eta_2}^k$ e quindi

$t \in R_p^k$

Output: s

MLWE: Esempio

$$p = 541,\ n = 4,\ k = 3,\ l = 2,\ \eta_1 = 3,\ \eta_2 = 2$$

$$R_{541} = \mathbb{Z}_{541}[x]/(x^4 + 1)$$

Matrice (selezionata random) $A \in R_{541}^{3\times 2}$:

$$A = \begin{bmatrix} 442 + 502x + 513x^2 + 15x^3 & 368 + 166x + 37x^2 + 135x^3 \\ 479 + 532x + 116x^2 + 41x^3 & 12 + 139x + 385x^2 + 409x^3 \\ 29 + 394x + 503x^2 + 389x^3 & 9 + 499x + 92x^2 + 254x^3 \end{bmatrix}$$

MLWE: Esempio

Segreto (selezionato random) $s \in S_3^2 \subset R_{541}^2$:

$$s = \begin{bmatrix} 2 - 2x + x^3 \\ 3 - 2x - 2x^2 - 2x^3 \end{bmatrix}$$

Errore (selezionato random) $e \in S_2^3 \subset R_{541}^3$:

$$e = \begin{bmatrix} 2 - 2x - x^2 \\ 1 + 2x + 2x^2 + x^3 \\ -2 - x^2 - 2x^3 \end{bmatrix}$$

MLWE: Esempio

Calcolo:

$$t = As + e = \begin{bmatrix} 30 + 252x + 401x^2 + 332x^3 \\ 247 + 350x + 259x^2 + 485x^3 \\ 534 + 234x + 137x^2 + 443x^3 \end{bmatrix} \quad \Rightarrow \quad \|t\|_\infty = 259$$

Problema:
Dato (A, t)
determinare un $s \in S_3^2$ e un $e \in S_2^3$
tali per cui $t = As + e$.

Versione decisionale di MLWE (D-MLWE)

Parametri del problema:

p primo

n, k, l interi con $k \geq l$

η_1, η_2 interi molto più piccoli di $p/2$

Input: (A, z) ottenuti come segue

$A \in_R R_p^{k \times l}$ (dove $R_p = \mathbb{Z}_p[x]/(x^n + 1)$)

$b \in_R \{0, 1\}$

$$z = \begin{cases} t = As + e, \ s \in_R S_{\eta_1}^l, \ e \in_R S_{\eta_2}^k & \text{se } b = 0 \\ t \in_R R_p^k & \text{se } b = 1 \end{cases}$$

Output: b

Complessità computazionale quantistica di MLWE e D-MLWE

Ad oggi non si conosce alcun algoritmo, né classico né quantistico, che risolva il problema MLWE o D-MLWE in tempo polinomiale.
La intrattabilità di questi problemi (anche utilizzando un modello di calcolo quantistico) si basa sulla complessità di alcuni problemi su reticoli:
SVP (Shortest Vector Problem)
CVP (Closest Vector Problem)
che sono notoriamente difficili anche per gli algoritmi quantistici.

Kyber-PKE

Kyber-PKE: generazione della chiave eseguita da Alice

1. $A \in_R R_q^{k \times k}$, $s \in_R S_{\eta_1}^k$, $e \in_R S_{\eta_2}^k$
2. $t = As + e$
3. Chiave pubblica:(A, t), Chiave privata:s

Nota: Calcolare s da A e t è un'istanza del problema MLWE.

Nota: In Kyber la matrice A è quadrata.

Kyber-PKE: Codifica eseguita da Bob

Messaggio $m \in \{0,1\}^n$

1. Ottenere da Alice la encryption key (A, t)
2. $m(x)$ polinomio calcolato da m
3. $r \in_R S_{\eta_1}^k,\ \ e_1 \in_R S_{\eta_2}^k,\ \ e_2 \in_R S_{\eta_2}$
4. $u = A^T r + e_1,\ \ v = t^T r + e_2 + \lceil \frac{p}{2} \rfloor m(x)$
5. $c = (u, v) \in R_p^k \times R_p$

Kyber-PKE: Deodifica eseguita da Alice

Ciphertext $c = (u, v)$

1. $m = Round_p(v - s^T u)$

Nota: s è la chiave privata di Alice.

Confronto con Elgamal

Bob	Eve	Alice
messaggio $= m$		$Prv = R \qquad Pub = g^R$
$g^R \leftarrow \cdots$	$\cdots \leftarrow g^R$	
genera k a caso		
	$g^k \rightarrow \cdots$	$\cdots \rightarrow g^k$
	$m(g^R)^k \rightarrow \cdots$	$\cdots \rightarrow m(g^R)^k$
		$m(g^R)^k((g^k)^R)^{-1} = m$

Bob	Eve	Alice
messaggio $= m$		$Prv = s \qquad Pub = (A, t)$
$(A, t) \leftarrow \cdots$	$\cdots \leftarrow (A, t)$	
genera r, e_1, e_2 a caso		
$u = A^T r + e_1$	$u \rightarrow \cdots$	$\cdots \rightarrow A^T r + e_1$
$v = t^T r + e_2 + \lceil \frac{p}{2} \rfloor m(x)$	$v \rightarrow \cdots$	$\cdots \rightarrow t^T r + e_2 + \lceil \frac{p}{2} \rfloor m(x)$
		$Round_p(v - s^T u) = m$

Kyber-PKE: Generazione delle chiavi (Alice)

Parametri: $p = 137,\ n = 4,\ k = 2,\ \eta_1 = 2,\ \eta_2 = 2$

$$A = \begin{bmatrix} 21 + 57x + 78x^2 + 43x^3 & 126 + 122x + 19x^2 + 125x^3 \\ 111 + 9x + 63x^2 + 33x^3 & 105 + 61x + 71x^2 + 64x^3 \end{bmatrix}$$

$$s = \begin{bmatrix} 1 + 2x - x^2 + 2x^3 \\ -x + 2x^3 \end{bmatrix} \qquad e = \begin{bmatrix} 1 - x^2 + x^3 \\ -x + x^2 \end{bmatrix}$$

Calcolo:

$$t = As + e = \begin{bmatrix} 55 + 96x + 123x^2 + 7x^3 \\ 32 + 27x + 127x^2 + 100x^3 \end{bmatrix}$$

Chiave pubblica: (A, t)

Chiave privata: s

Kyber-PKE: Codifica (Bob)

Messaggio: $m = 0111 \quad \Rightarrow \quad m(x) = x + x^2 + x^3$

Scelte casuali:

$$r = \begin{bmatrix} -2 + 2x + x^2 - x^3 \\ -1 + x + x^2 \end{bmatrix}$$

$$e_1 = \begin{bmatrix} 1 - 2x^2 + x^3 \\ -1 + 2x - 2x^2 + x^3 \end{bmatrix} \quad e_2 = 2 + 2x - x^2 + x^3$$

Calcoli:

$$u = A^T r + e_1 = \begin{bmatrix} 56 + 32x + 77x^2 + 9x^3 \\ 45 + 21x + 2x^2 + 127x^3 \end{bmatrix}$$

$$v = t^T r + e_2 + \left\lceil \frac{p}{2} \right\rceil m(x) = 3 + 10x + 8x^2 + 123x^3$$

Ciphertext: $c = (u, v)$

Kyber-PKE: Decodifica (Alice)

Calcolo:

$$v - s^T u = 4 + 60x + 79x^2 + 66x^3$$

Dopo l'arrotondamento: $m(x) = x + x^2 + x^3$ e quindi $m = 0111$.

Kyber-PKE: Sicurezza da chosen-plaintext attack

Kyber-PKE è IND-CPA sicuro (Indistinguishability under Chosen-Plaintext Attack), assumendo che il problema D-MLWE sia intrattabile.

L'operazione di cifratura può essere scritta come:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} A^T \\ t^T \end{bmatrix} r + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \lceil \frac{p}{2} \rfloor m \end{bmatrix}$$

Sotto l'assunzione D-MLWE intrattabile, $A^T r + e_1$ è indistinguibile da casuale.
Anche $t^T r + e_2$ è indistinguibile da casuale.
Quindi, l'avversario vede un valore casuale + messaggio mascherato $\rightarrow$ impossibile ottenere informazioni su m.

Kyber-PKE: la decodifica non sempre è corretta

La decodifica produce correttamente m? ovvero
$m = \text{Round}_q(v - s^T u)$?

$$
\begin{aligned}
v - s^T u &= \left(t^T r + e_2 + \left\lceil \frac{p}{2} \right\rfloor m \right) - s^T (A^T r + e_1) \\
&= s^T A^T r + e^T r + e_2 + \left\lceil \frac{p}{2} \right\rfloor m - s^T A^T r - s^T e_1 \\
&= e^T r + e_2 - s^T e_1 + \left\lceil \frac{p}{2} \right\rfloor m
\end{aligned}
$$

$$E(x) = e^T r + e_2 - s^T e_1$$

La decifratura funziona se ogni coefficiente E_i soddisfa:

$$-p/4 < E_i \ mods \ p < p/4 \quad \Rightarrow \|E\|_\infty < p/4$$

Kyber-PKE: la decodifica non sempre è corretta

$$\|E\|_\infty \leq kn\eta_1\eta_2 + \eta_2 + kn\eta_1\eta_2$$

Nel caso di ML-KEM-768:

$$p = 3329,\ n = 256,\ k = 3,\ \eta_1 = \eta_2 = 2 \Rightarrow \|E\|_\infty \leq 6146 > p/4$$

Conclusione: la decodifica non è garantita, ma:

$$\|E\|_\infty < p/4 \quad \text{con probabilità estremamente alta.}$$

Kyber-KEM

PKE (Public-Key Encryption)Vs KEM (Key Encapsulation Mechanism)

	PKE	**KEM**
Obiettivo	Cifrare un messaggio	Condividere una chiave segreta
Input mittente	Messaggio m	Nessun input
Output mittente	Ciphertext c contenente m	Ciphertext + chiave K
Output destinatario	m	K
Sicurezza tipica	IND-CPA o IND-CCA	IND-CCA
Esempi	RSA, ElGamal, Kyber-PKE	Kyber-KEM, NTRU-KEM

Nozioni di sicurezza: IND-CPA vs IND-CCA

IND = Indistinguishability
CPA = Chosen Plaintext Attack
CCA = Chosen Ciphertext Attack

Nozioni di sicurezza: IND-CPA vs IND-CCA

- IND-CPA: L'avversario può scegliere e cifrare liberamente messaggi. Poi chiede di cifrare m_0 e m_1 (sempre scelti da lui), riceve c che è il chipertext di m_0 o di m_1. Se riesce a capire quale dei due è stato cifrato, il cifrario non è IND-CPA.
- IND-CCA: L'avversario può scegliere e cifrare liberamente messaggi. Poi chiede di cifrare m_0 e m_1 (sempre scelti da lui), riceve c che è il chipertext di m_0 o di m_1. L'avversario può scegliere e decifrare liberamente qualunque ciphertext tranne c. Se riesce a capire quale dei due è stato cifrato, il cifrario non è IND-CCA.
- Differenza chiave: in IND-CCA l'avversario ha accesso a un oracolo di decifratura, rendendo la sfida più difficile da vincere per lo schema.

Nozioni di sicurezza: IND-CPA vs IND-CCA

Gerarchia di sicurezza:

$$IND\text{-}CCA \Rightarrow IND\text{-}CPA \Rightarrow \text{semantica}$$

Un cifrario è semanticamente sicuro se un avversario non può ricavare nessuna informazione significativa dal ciphertext, neppure una singola proprietà del messaggio.

Trasformazione di Fujisaki-Okamoto (FO)

FO trasforma uno schema IND-CPA (come Kyber-PKE) sicuro contro attacchi a testo cifrato scelto (IND-CCA).

Gli schemi PKE (Public-Key Encryption) sono di solito solo IND-CPA sicuri. Per l'uso pratico (es. TLS) serve invece la sicurezza IND-CCA.

FO Transform (1999): Prende uno schema IND-CPA. Usa hash e re-encryption per verificare la coerenza. Se il decapsulamento non "ricostruisce" correttamente, viene rigettata.

Intuizione: Il ciphertext dipende in modo deterministico dal messaggio segreto. Durante il decapsulamento si controlla che il ciphertext sia coerente e quindi impedisce attacchi attivi.

Kyber-KEM

Kyber-KEM è un quantum-safe Key Encapsulation Mechanism.

Kyber-KEM è stato standardizzato dal NIST in FIPS 203 (agosto 2024) dove è chiamato ML-KEM (Module-Lattice-based KEM).

Kyber-KEM è stato progettato applicando la trasformazione di Fujisaki-Okamoto a Kyber-PKE che è un Public-key Encryption Scheme.

Kyber-KEM

Kyber è stato sviluppato da un team internazionale di crittografi, tra i quali: Daniel J. Bernstein, Johannes Buchmann, Léo Ducas, Eike Kiltz, Tanja Lange, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, Damien Stehlé, Johannes Albrecht, Christian Badertscher.

ML-KEM-768

ML-KEM-768 è una istanza specifica di Kyber-KEM che utilizza i seguenti parametri:

Parametro	**Valore**
Primo p	3329
Grado del polinomio n	256
Dimensione del modulo k	3
Ampiezza del segreto η_1	2
Ampiezza dell'errore η_2	2

KEM: Key Encapsulation Mechanisms

Un KEM consente a due parti di stabilire una chiave segreta condivisa tramite crittografia a chiave pubblica.

Un KEM è composto da tre algoritmi:

Key Generation. Ogni utente (es. Alice) esegue questo algoritmo per generare una chiave pubblica di incapsulamento ek e una chiave privata di decapsulamento dk

Encapsulation. L'altra parte (es. Bob) usa ek per generare una chiave segreta condivisa K e un ciphertext c. Bob invia c ad Alice.

Decapsulation. Alice usa la chiave privata dk per derivare la stessa chiave K a partire da c.

Kyber-KEM e la trasformazione Fujisaki-Okamoto

Per applicare la trasformazione Fujisaki-Okamoto a Kyber-PKE occorre utilizzare tre funzioni hash:

- $G : \{0,1\}^* \rightarrow \{0,1\}^{512}$
- $H : \{0,1\}^* \rightarrow \{0,1\}^{256}$
- $J : \{0,1\}^* \rightarrow \{0,1\}^{256}$

G, H e J sono funzioni hash SHA-3.

Resistenza quantistica di SHA-3

SHA-3 è parzialmente resistente ad attacchi quantistici.

Funzione	Impronta	Sicurezza effettiva
SHA3-256	256 bit	Accettabile
SHA3-512	512 bit	Robusta
SHAKE-128	variabile	Debole
SHAKE-256	variabile	Standard per PQ

Conclusione: SHA-3 è quantum-resilient, ma per piena sicurezza post-quantistica servono primitive basate su problemi resistenti ad attacchi quantistici.

Kyber-KEM: Generazione delle chiavi (Alice)

1. Esegue l'algoritmo di generazione delle chiavi descritto in Kyber-PKE:
   Chiave pubblica: $ek = (A, t)$
   Chiave segreta: s
2. $z \in_R \{0,1\}^{256}$
3. Encapsulation key: $ek = (A, t)$
   Decapsulation key: $dk = (s, ek, H(ek), z)$

Kyber-KEM: Incapsulamento (Bob)

1. Ottiene la chiave pubblica di Alice: ek
2. $m \in_R \{0,1\}^{256}$
3. $h = H(ek)$
4. $(K, R) = G(m, h)$
5. Usa R per cifrare m con Kyber-PKE e ottiene c. Le quantità random che servono in Kyber-PKE vengono generate a partire da R (derandomization)
6. Chiave segreta $= K$
7. Spedisce c a Alice

Kyber-KEM: Decapsulamento (Alice)

1. Calcola m' decriptando c (usa Kyber-PKE e la chiave segreta s)
2. $(K', R') = G(m', H(ek))$
3. Calcola c' codificando nuovamente m' usando R'
4. Se $c = c'$, allora $K = K'$
5. Altrimenti return $K = J(z, c)$

Kyber-KEM: trasformazione FO

Encapsulamento e derandomizzazione:

Si seleziona $m \in \{0,1\}^{256}$ casuale.
Si calcolano $h = H(ek)$ e $(K, R) = G(m, h)$.
I polinomi (r, e_1, e_2) usati per cifrare m sono derivati da R e non sono scelti a caso

Kyber-KEM: trasformazione FO

Decapsulamento:

Si decifra c per ottenere m'.
Si ricalcolano $(K', R') = G(m', H(ek))$.
Si rigenera c' e si confronta con c.
Se $c = c'$, allora $K = K'$, altrimenti
$K = J(z, c)$ (random e quindi totalmente indipendente da K).

Kyber-KEM: sicurezza

In Kyber-KEM il decapsulamento produce K se e solo se chi ha effettuato l'incapsulamento conosceva già K. Questo rende Kyber-KEM resistente agli attacchi di tipo CCA

Kyber-KEM: fallimento del decapsulamento

- Il decapsulamento fallisce quando il ciphertext rigenerato c' è diverso da quello ricevuto c
- In tal caso, la chiave prodotta $K = J(z, c)$ sarà quasi certamente diversa da quella incapsulata.
- Questo può avvenire anche se Alice e Bob sono "onesti", a causa di una piccola probabilità di errore in Kyber-PKE
- Il tasso di fallimento nel decapsulamento è dimostrabilmente trascurabile.

Kyber-KEM: conclusione

Assumendo che il problema D-MLWE sia intrattabile e che le funzioni hash G, H, J siano funzioni random, Kyber-KEM è IND-CCA sicuro (indistinguibilità sotto attacco a ciphertext scelto)

Assumendo che gli attaccanti possono effettuare query classiche e query quantistiche su G, H e J, Kyber-KEM resiste anche ad attaccanti quantistici

Perché Kyber-KEM è resistente agli attacchi CCA?

Trasformazione Fujisaki-Okamoto rende la cifratura deterministica (via hash) e abilita il controllo del ciphertext durante il decapsulamento.
Il messaggio decriptato viene usato per ricostruire il ciphertext originale. Se $c \neq c'$, viene restituita una chiave casuale: nessuna informazione utile per l'attaccante
Nessun oracolo di decifratura utile. Anche se l'attaccante può manipolare ciphertext, non può distinguerli né ottenere vantaggi concreti.
Dimostrazione formale nel modello ROM: la sicurezza IND-CCA di Kyber-KEM è dimostrata nel Random Oracle Model, assumendo che D-MLWE sia difficile.

Perché usare Kyber-KEM invece di Kyber-PKE diretto?

- Sicurezza più forte (IND-CCA)
  Kyber-KEM è sicuro anche contro attacchi attivi, grazie alla trasformazione Fujisaki-Okamoto. Kyber-PKE è solo IND-CPA.
- Cifratura ibrida pronta all'uso
  Kyber-KEM genera una chiave segreta K da usare con AES o ChaCha20. Ideale per protocolli come TLS, Signal, WireGuard.
- Nessun messaggio diretto da cifrare
  Non serve gestire padding, lunghezza o formato dei messaggi.
- Più robusto contro RNG deboli
  Kyber-KEM è derandomizzato via hash: evita problemi storici legati a cifratura con randomness debole (es. RSA).
- Modularità
  Separazione tra *negoziazione della chiave* (KEM) e *cifratura dei dati* (AEAD). Favorisce l'integrazione sicura nei protocolli.

Parametri di sicurezza: Kyber-KEM

	Categoria NIST	Sicurezza equivalente
ML-KEM-512	Categoria 1	128-bit (AES-128)
ML-KEM-768	Categoria 3	192-bit (AES-192)
ML-KEM-1024	Categoria 5	256-bit (AES-256)

Le categorie 1, 3 e 5 corrispondono alla resistenza contro attacchi quantistici brute force su cifrari a 128, 192 e 256 bit.

Parametri di sicurezza: Kyber-KEM

	512	768	1024
Categoria NIST	1	3	5
k (dimensione modulo)	2	3	4
p (modulo)	3329	3329	3329
n (grado dei polinomi)	256	256	256
η_1 (errore secret)	3	2	2
η_2 (errore noise)	2	2	2
Plaintext incapsulato	32 B	32 B	32 B
Chiave pubblica	800 B	1184 B	1568 B
Chiave segreta	1632 B	2400 B	3168 B
Ciphertext	736 B	1088 B	1440 B
Failure rate	$\leq 2^{-129}$	$\leq 2^{-151}$	$\leq 2^{-189}$